Perfect simulation and non-monotone Markovian systems*

Ana Bušić
INRIA Grenoble - Rhône-Alpes
51, Av. J. Kuntzmann, 38330
Montbonnot, France
Ana.Busic@imag.fr

Bruno Gaujal
INRIA Grenoble - Rhône-Alpes
51, Av. J. Kuntzmann, 38330
Montbonnot, France
Bruno.Gaujal@inria.fr

Jean-Marc Vincent
Université Joseph Fourier
51, Av. J. Kuntzmann, 38330
Montbonnot, France
Jean-Marc.Vincent@imag.fr

ABSTRACT

Perfect simulation, or coupling from the past, is an efficient technique for sampling the steady state of monotone discrete time Markov chains. Indeed, one only needs to consider two trajectories corresponding to minimal and maximal state in the system. We show here that even for non-monotone systems one only needs to compute two trajectories: an infimum and supremum envelope. Since the sequence of states obtained by taking infimum (resp. supremum) at each time step does not correspond to a feasible trajectory of the system, envelopes and not feasible trajectories. We show that the envelope approach is efficient for some classes of non-monotone queuing networks, such as networks of queues with batch arrivals, queues with fork and join nodes and/or with negative customers.

Categories and Subject Descriptors

I.6 [**Computing Methodologies**]: Simulation and modeling; G.3 [**Probability and Statistics**]: Markov processes

Keywords

Markov chains, perfect simulation, queuing networks

1. INTRODUCTION

Simulation approaches are alternative methods to estimate the stationary behavior of stochastic systems by providing samples distributed according to the stationary distribution, even when it is impossible to compute this distribution numerically. Propp and Wilson used a backward coupling [6] to derive a simulation algorithm (PSA: Perfect Simulation Algorithm) providing perfect sampling (*i.e.* which distribution is stationary) of the state of a *discrete time Markov chain* over a finite state space $\mathcal{X}$. The main idea is to start many simulated trajectories from all $x \in \mathcal{X}$ at some time in the past until time $t = 0$. If the output is the same for all trajectories, then the chain has coupled and the common state has the stationary distribution of the Markov chain. Otherwise, the simulations are started further in the past.

*This work was partially supported by ANR-05-BLAN-0009-02 SMS and ANR-06-SETIN-002 Checkbound.

ValueTools 2008, October 21 - 23, 2008, Athens, GREECE.

Although the Perfect Simulation Algorithm provides exact samples for all recurrent finite Markov chains, its complexity can be high. It suffers from two drawbacks that jeopardize its applicability for very large chains. The first one is the fact that the coupling time can be very large. Some work focused on the computation of the coupling time for certain classes of Markov chains. It was shown in [3] that Markov chains modelling a class of networks of queues with finite capacities have a very small coupling time with respect to the size of the state space.

The second factor in the complexity of PSA is the fact that one needs to run one simulation per state in $\mathcal{X}$. When the state space is ordered and the Markov chain is monotone Propp and Wilson have shown that it is enough to run two simulations only: one starting from the largest state and one from the smallest state. Since then, several extensions were proposed when the state space is partially ordered. In that case one needs to run one simulation per extremal state. This is the case for stochastic event graphs, a sub-class of stochastic Petri nets [2], as well as for heaps of pieces [1]. However, when the Markov chain is not monotone, nothing has been proposed so far to avoid the construction of one trajectory per state in $\mathcal{X}$. Meanwhile, non-monotone chains are very common. In the context of networks of queues, introducing fork and join nodes or negative customers or batch arrivals and services destroys the monotonicity of the underlying Markov chain.

The goal of this paper is to provide a new perfect simulation algorithm (EPSA: Envelope Perfect Simulation Algorithm) that only uses two simulated trajectories for general non-monotone Markov chains. This is done by computing from the past, a lower and an upper envelope of all trajectories of the Markov chain. Whenever the two envelopes couple, the coupling state is distributed according to the stationary distribution of the chain. This new simulation technique raises new questions. Do envelopes couple with probability one? Actually, using the construction of envelopes given in section 3, they do not couple in some cases. However, we show in §5.2 that it is always possible to change the constructive definition of the Markov chain (*i.e.* the function Φ) so that the corresponding envelopes will couple. However, the complexity of this construction is quadratic in the size of the state space.

If envelopes couple then the EPSA provides perfect samples for the Markov chain but its complexity can make it impossible to use in practice. This question is discussed in details in the paper. We show that in some cases, the coupling

time of the envelopes is similar to the coupling time of the chain (in § 4.1 and §4.2) while in other cases, it becomes embarrassingly larger (in §4.3).

When envelopes do not couple, it is still possible to exploit them because they provide bounds on the stationary distribution (see § 5) or because they can serve as an initial step in the perfect simulation: use envelopes as long as they contract the state space and continue with an exhaustive simulation from that point on (this is called splitting in the paper, see § 5).

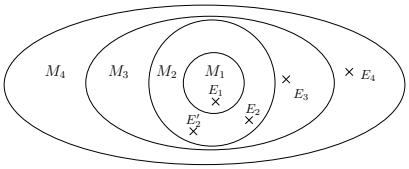

Figure 1: Markov chains and perfect simulation.

All this extends the limit of applicability of perfect simulation beyond the monotone case, as illustrated in Figure 1. The set M_1 is the set of monotone Markov chains, where perfect simulation was known to be very efficient. The set M_2 is the set of non-monotone Markov chains, where envelope perfect simulation can be used efficiently (the coupling time is similar to the coupling time of the chain). The set M_3 is the set of all Markov chains where the envelopes do couple but take a much larger time to couple than the Markov chain, while the set M_4 is the set where the envelopes do not couple so that splitting must be used to get samples (or a new set of events must be constructed to guarantee that envelopes will couple).

Another contribution of this paper is to provide concrete examples in the context of queuing networks falling in each of these sets. The example E_1 is any network of finite queues with monotone routing. Example E_2 is a network as E_1 with negative customers (the arrival of a negative customer in a queue destroys one regular customer). Example E_2' is a network as E_1 with fork and join nodes (a fork splits a customer into two independent customers while a join replaces two customers by a single one). Example E_3 is a network of queues with individual customers as well as batches (customers may arrive by batches and may also be served and routed alone). Finally, example E_4 is a network of queues with only batches larger than two. All these experiments have been done with the perfect simulation software Ψ^2 [10], freely available at `http://psi.gforge.inria.fr`.

2. PERFECT SIMULATION

Let $\{X_n\}_{n\in\mathbb{N}}$ be an irreducible and aperiodic discrete time Markov chain with a finite state space $\mathcal{X}$ and a transition matrix $P = (p_{i,j})$. Let π denote the steady state distribution of the chain: $\pi = \pi P$. The evolution of the Markov chain can always be described by a stochastic recurrence sequence $X_{n+1} = \Phi(X_n, e_{n+1})$, with $\{e_n\}_{n\in\mathbb{Z}}$ an independent and identically distributed sequence of events $e_n \in \mathcal{E}$, $n \in \mathbb{N}$. The transition function $\Phi : \mathcal{X} \times \mathcal{E} \to \mathcal{X}$ verifies the property that $\mathbb{P}(\Phi(i, e) = j) = p_{i,j}$ for every pair of states $(i, j) \in \mathcal{X} \times \mathcal{X}$ and a random event $e \in \mathcal{E}$.

Let $\Phi^n : \mathcal{X}\times\mathcal{E}^n \to \mathcal{X}$ denote the function whose output is the state of the chain after n iterations and starting in state $x \in \mathcal{X}$. That is, $\Phi^n(x, e_{1\to n}) \stackrel{\text{def}}{=} \Phi(\ldots\Phi(\Phi(x, e_1), e_2), \ldots, e_n)$. This notation can be extended to sets of states: for $A \subset \mathcal{X}$, $\Phi^n(A, e_{1\to n}) \stackrel{\text{def}}{=} \{\Phi^n(x, e_{1\to n}), x \in A\}$. In the following, $|A|$ denotes the cardinality of set A.

THEOREM 1 ([6]). *There exists $\ell \in \mathbb{N}$ such that*

$$\lim_{n\to\infty} \left|\Phi^n(\mathcal{X}, e_{-n+1\to 0})\right| = \ell \text{ almost surely.}$$

The system couples if $\ell = 1$. In that case, the value of $\Phi^n(\mathcal{X}, e_{-n+1\to 0})$ is steady state distributed.

Algorithm 1: Perfect Simulation Algorithm (PSA)

Data: A function Φ and i.i.d. events $\{e_{-n}\}_{n\in\mathbb{N}}$
Result: One state in $\mathcal{X}$ generated according to the stationary distribution of the system

```
begin
    n := 1;
    foreach state x ∈ X do
        y(x) := x;
    repeat
        foreach state x ∈ X do z(x) := y(Φ(x, e_{-n}));
        foreach state x ∈ X do y(x) := z(x);
        n := n + 1;
    until |{y(x), x ∈ X}| = 1 ;
    return {y(x), x ∈ X};
end
```

Theorem 1 has an algorithmic counterpart (PSA), given in Algorithm 1. The main drawback of PSA algorithm is the fact that one needs to simulate the MC starting with all states in $\mathcal{X}$, which could be too large for Algorithm 1 to be used in practice.

Several approaches have been used to overcome this problem. The main one is already present in [6]. When the state space $\mathcal{X}$ is partially ordered and when the function $\Phi(\cdot, e)$ is monotone for all e, then the Markov chain is monotone and it is possible to generate a steady state by starting Algorithm 1 with maximal and minimal states only. This technique has been successfully used in [7] to construct PSA for network of queues. When $\Phi(\cdot, e)$ is not monotone, one can still use monotone bounds, as in [4].

3. ENVELOPES

The approach proposed here generalizes what has been done in [5] to simulate non-monotone Markov chains. Its main advantage is that it does not need any preliminary assumption on the structure of the Markov chain.

Assume that the state space $\mathcal{X}$ is equipped with an lattice order relation $\leq$ and consider a new transition function Γ, called the envelope, defined on the Cartesian product of the state space :

$$\Gamma : \mathcal{X} \times \mathcal{X} \times \mathcal{E} \to \mathcal{X} \times \mathcal{X},$$

for all m, M in $\mathcal{X}$ and e in $\mathcal{E}$,

$$\Gamma_1(M, m, e) \stackrel{\text{def}}{=} \overline{\Phi}(M, m, e) \stackrel{\text{def}}{=} \sup_{m\leq x\leq M} \Phi(x, e)$$

$$\Gamma_2(M, m, e) \stackrel{\text{def}}{=} \underline{\Phi}(M, m, e) \stackrel{\text{def}}{=} \inf_{m\leq x\leq M} \Phi(x, e).$$

Let us call $T \stackrel{\text{def}}{=} \sup \mathcal{X}$ (resp . $B \stackrel{\text{def}}{=} \inf \mathcal{X}$) the top (resp. bottom) element of $\mathcal{X}$. The process

$$(M_n, m_n) \stackrel{\text{def}}{=} \Gamma^n(T, B, e_{1\to n})$$

is a Markov chain over the state space $\mathcal{X} \times \mathcal{X}$. However, the upper envelope alone is not a Markov chain (it depends on the lower enveloppe), neither is the lower one.

THEOREM 2. *Assume that (M_n, m_n) hits the diagonal $\mathcal{D}$ (i.e. states of the form (x, x)) in finite time:*

$$\tau_e \stackrel{\text{def}}{=} \min\left\{n : \Gamma^n(T, B, e_{-n+1\to 0}) \in \mathcal{D}\right\},$$

then τ_e is a backward coupling time of the initial Markov chain X_n. The state defined by $\Gamma^{\tau_e}(T, B, e_{-\tau_e+1\to 0})$ has the steady state distribution of X_n.

PROOF. The proof simply uses the fact that if $(M_0, m_0) \stackrel{\text{def}}{=} \Gamma^n(T, B, e_{-n+1\to 0})$, then the set $\Phi^n(\mathcal{X}, e_{-n+1\to 0})$ is included in the set $\{x : m_0 \le x \le M_0\}$, consequently if the latter is reduced to one point, so is the set $\Phi^n(\mathcal{X}, e_{-n+1\to 0})$ and the backward scheme has coupled. □

Now, Algorithm 1 can be adapted to the envelope simulation: start the simulation with only two states: T and B and iterate using the bi-dimensional function $(\overline{\Phi}, \underline{\Phi})$.

Algorithm 2: Backward simulation algorithm using envelopes (EPSA)

Data: - Φ , $\{e_{-n}\}_{n\in\mathbb{N}}$
- Γ the pre-computed envelope function
Result: A state $x^* \in \mathcal{X}$ generated according to the stationary distribution of the system

```
begin
    n = 1; M := T; m := B;
    repeat
        for i = n downto 1 do
            (M, m) := Γ(M, m, e_{-i+1});
        n := 2n;
    until M = m ;
    return M;
end
```

This new algorithm (Algorithm 2) calls for several comments.
1- Everything works the same if $\overline{\Phi}$ (resp. $\underline{\Phi}$) is replaced by an upper (resp. lower) bound on the supremum (res. infimum).

2- The definition of the envelopes is based on the constructive definition Φ of the Markov chain. If one comes up with a new construction Φ' of the Markov chain, then the envelopes are modified accordingly.

3- If the function $\Phi(., e)$ is non-decreasing for all event e, then for any $m \le M$, $\overline{\Phi}(M, m, e) = \Phi(M, e)$ and $\underline{\Phi}(M, m, e) = \Phi(m, e)$, so that Algorithm EPSA coincides with the classical monotone perfect simulation algorithm for monotone Markov chains.

4- The envelope approach may not gain over the general PSA because of two problems:

(P_1) The assumption that (M_n, m_n) hits the diagonal in Theorem 2 may not hold.

(P_2) Even if Theorem 2 applies, the complexity of the algorithm may be prohibitive. The average complexity of EPSA is $O(\mathcal{C}(\overline{\Phi}, \underline{\Phi}) \cdot \mathbb{E}\tau_e)$ where τ_e is the number of iterations of the main loop of EPSA (called the coupling time in the following) and $\mathcal{C}(\overline{\Phi}, \underline{\Phi})$ is the complexity (number of elementary operations) to compute $\overline{\Phi}$ and $\underline{\Phi}$. Meanwhile, the average complexity of the classical PSA is $O(\mathcal{C}(\Phi) \cdot |\mathcal{X}| \cdot \mathbb{E}\tau)$, where τ is the coupling time of PSA, $|\mathcal{X}|$ the size of the state space and $\mathcal{C}(\Phi)$ the complexity of the computation of Φ.

If $\mathcal{C}(\overline{\Phi}, \underline{\Phi})$ is not too high w.r.t $\mathcal{C}(\Phi)$ (this is usually the case in the following examples), then the comparison of the two methods is essentially a comparison between the coupling time of EPSA ($\mathbb{E}\tau_e$) and ($|\mathcal{X}| \cdot \mathbb{E}\tau$).

In the following we will focus on problems P_1 and P_2 in the context of networks of queues. We show that networks of queues with negative customers are not monotone but envelopes couple with probability one (problem P_1) and the coupling of the envelope algorithm (P_2) remains reasonable (see §4.1). The same holds when fork and join nodes are introduced (see § 4.2). In these cases, there is a clear gain to use EPSA over PSA (the complexity is divided by the size of the state space).
However, if customers arrive by batches, things change dramatically, as shown in § 4.3. First, envelopes never couple in some cases. We provide a characterization of the set of networks of queues with batches that do couple with probability one. When coupling occurs, the complexity $\mathcal{C}(\overline{\Phi}, \underline{\Phi})$ of computing the envelopes remains small but the coupling time $\mathbb{E}\tau_e$ can grow exponentially with the parameters of the network. We can bypass this by using a split (see § 5) where coupling occurs w.p.1 and the coupling time remains similar to the coupling time of PSA. Again, the complexity is divided by the size of the state space in most cases.

4. EPSA FOR NETWORKS OF QUEUES

We consider an open network $\mathcal{N}$ consisting of N queues $Q_1, \ldots, Q_N$. Each queue Q_i has a finite capacity, denoted by C_i, $i = 1, \ldots N$ and a finite number of servers. The state space of a single queue Q_i is $\mathcal{X}_i = \{0, \ldots C_i\}$ and the state space of the network is $\mathcal{X} \stackrel{\text{def}}{=} (\mathcal{X}_1 \times \cdots \times \mathcal{X}_N)$. $\mathcal{X}$ is a lattice for the usual product order, denoted $\le$. Its Bottom (resp. Top) element is $(0, \ldots, 0)$ (resp. $(C_1 \ldots, C_N)$). In the following we will denote by 1_i the vector with null components except the ith one which is equal to one.

Since the network is open, customers are able to enter and depart the network. The customers who enter from outside the network arrive according to a Poisson process. Also, the service time at any server is exponentially distributed. After finishing his service at a server, a customer is either directed to another queue or leaves the network according to a routing policy. Here, we consider a quite general class of routing policies, based on indexes (it includes the classical routing policies such as deterministic, join the shortest queue, priority routing and others, see [7]). The set of output queues of a queue Q_i is denoted Δ_i. It may include the dummy queue Q_{N+1} corresponding to the exit of the network. Also, each queue Q_i is equipped with a family I^i of index functions,

one for each queue in Δ_i: $I_k^i : \{0, \dots, C_k\} \to \mathbb{R}$. The routing works as follows: a customer ending its service at Q_i in state $(x_1, \dots, x_N)$ is sent to queue Q_j with

$$j = \arg\min_{k \in \Delta_i} I_k^i(x_k).$$

Under these assumptions the network evolves as a finite discrete time Markov chain, after uniformization. It can be constructed using events r_i that correspond to the transfer of a customer after a service at queue Q_i (for $i \leq N$) or to exogenous arrivals (for $i = N+1$). The transition function of event r_i, $1 \leq i \leq N$, is given by:

$$\Phi(x, r_i) = x - 1_i \delta_{x_i > 0} + 1_j \delta_{x_i > 0}\ \delta_{x_j < C_j},$$

where $j = \arg\min_{k \in \Delta_i} I_k^i(x_k)$ and δ_E is the indicator function of set E. The transition function of event r_{N+1}, corresponding to exogenous arrivals, is given similarly by:

$$\Phi(x, r_{N+1}) = x + 1_j\ \delta_{x_j < C_j},$$

where $j = \arg\min_{k \in \Delta_{N+1}} I_k^{N+1}(x_k)$, Δ_{N+1} is the set of entrance queues of the network and I^{N+1} is the family of index functions for the dummy queue.

All index functions are assumed to be non-decreasing. In that case, it is shown in [9] that the Markov chain of the network is monotone.

If the index functions can be computed in constant time (for example, they are given as arrays), then the complexity of the computation of $\Phi(., e)$ is $\mathcal{C}(\Phi) = O(N)$. Indeed, this complexity boils down to computing the minimum of at most N values (to obtain the minimum index).

As for EPSA, the envelopes correspond to the trajectories of the Markov chain starting with the top (resp. bottom) element of the state space because of monotonicity of the chain. This shows that the complexity of computing the envelopes is the same as the complexity of computing Φ: $\mathcal{C}(\overline{\Phi}, \underline{\Phi}) = 2\,\mathcal{C}(\Phi)$. Also the coupling time of EPSA and the coupling time of PSA have the same distribution: $\tau_e =_d \tau$. Therefore, the complexity of EPSA is $O(N \cdot \mathbb{E}\tau)$ and the complexity of PSA is $O(N \cdot |\mathcal{X}| \cdot \mathbb{E}\tau)$, with a clear gain for EPSA. Actually, in this case, EPSA corresponds to the well known simulation of monotone Markov chains.

4.1 Negative customers

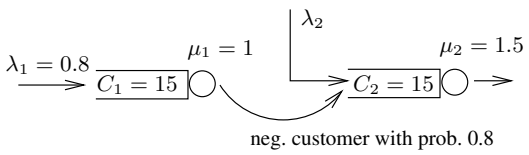

Figure 2: A network with negative customers.

Here, we consider a network with additional events (g_i): negative customers. The arrival of a negative customer in a queue kills one customer in that queue, if any. The evolution of the state under the transfer of a negative customer from Q_i is $\Phi(x, g_i) = x - 1_i \delta_{x_i > 0} - 1_j \delta_{x_i > 0}\ \delta_{x_j > 0}$ where $j = \arg\min_{k \in \Delta_i} I_k^i(x_k)$. In that case, the Markov chain is no longer monotone. Indeed, consider two queues in tandem Q_1, Q_2 with a negative customer leaving Q_1 to join Q_2 under two states $x = (0, 1) < x' = (1, 1)$. Then $\Phi(x, g_1) = (0, 1) > \Phi(x', g_1) = (0, 0)$.

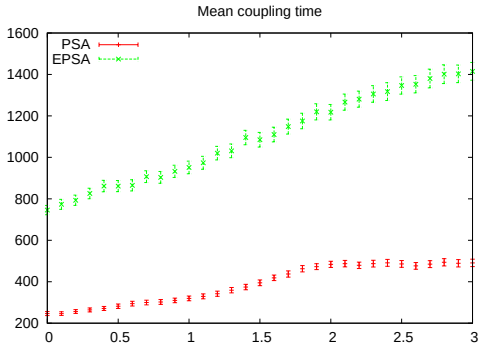

Figure 3: Coupling times of PSA and EPSA algorithms for the network in Figure 2 as a function of λ_2.

The computation of envelopes $\overline{\Phi}(M, m, g_i), \underline{\Phi}(M, m, g_i)$ can be cumbersome because of the value of the argmin of all states between m and M. However, tight upper and lower bounds (reached for some index routings) are easy to compute with complexity $O(N)$. For positive customers transfers, the envelopes are the same as before: $\overline{\Phi}(M, m, t_i) = \Phi(M, t_i)$ and $\underline{\Phi}(M, m, t_i) = \Phi(m, t_i)$. As for a transfer of a negative customer from queue Q_i, if $m_i = 0$ and $M_i > 0$, then:

$$\begin{aligned} \overline{\Phi}(M, m, g_i) &\leq M - 1_i, \\ \underline{\Phi}(M, m, g_i) &\geq m - \sum_{j \in \Delta_j} 1_j \delta_{m_j > 0}, \end{aligned}$$

otherwise:

$$\overline{\Phi}(M, m, g_i) = \Phi(M, g_i), \quad \underline{\Phi}(M, m, g_i) = \Phi(m, g_i).$$

As for the coupling time, it is easy to show that the bounds on the envelopes computed above always couple (the state where both bounds are equal to 0 can be reached with positive probability). However, the coupling time of these bounds is larger than the coupling time of PSA: $\tau \leq_{st} \tau_e$. However, the numerical experiments reported here (as well as many others) suggest that τ and τ_e have the same order of magnitude. We ran simulations over the small network displayed in Figure 2. The coupling time of PSA and EPSA are reported in Figure 3 (in the following all data are given with 95 % confidence intervals). The coupling time of EPSA is around 3 times larger than PSA. This means that EPSA is about 40 times faster than PSA in this case.
We also ran simulations of the same network without the negative customers. We compared the coupling time for the same average number of customers in queue 2 (Figure 4). They remain in the same order of magnitude. This shows that EPSA is not only much faster than PSA but also very efficient in absolute terms: the simulation time is similar to that of a similar monotone system.

4.2 Fork and join

Here we consider a network with additional types of queues: forks and joins. A fork queue is such that at service completion, several customers are sent simultaneously to several disjoint output queues if none is full, otherwise they are all lost. A join is a queue with several buffers. The server is only active when no buffer is empty. A service completion

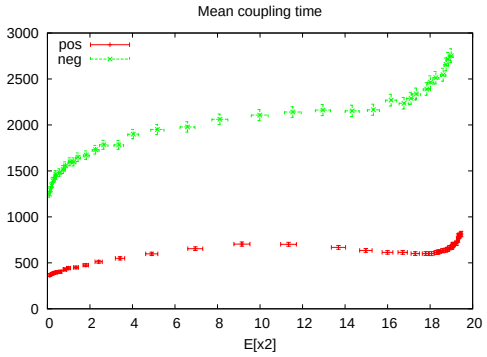

Figure 4: Coupling times of EPSA algorithm for a network with ("neg") and without ("pos") negative customers.

removes one customer in each buffer. Here again, the addition of a fork node (or a join node) in the queuing network makes it non-monotone.

Unlike for negative customers the computation of the exact envelopes is easy and is in $O(N)$ because fork and join nodes no do interfere with routing (not detailed). EPSA couples in finite time w.p.1 and experimental results show that the coupling times τ_e and τ are very close to each other (even closer than for negative customers).

4.3 Batch arrivals

Here, we consider a system of N queues with finite capacities C_i, $1 \leq i \leq N$, with batch arrivals (made of several customers) and atomic rejections: if all the customers forming a batch cannot be accepted all together into the destination queue, then the whole batch is rejected. We have the following events in the system:

An arrival of batch of size k, $1 \leq k \leq K$, where K is the maximal size of a batch and such that, in state x, the destination queue is given by $j = \arg\min_i I_i(x_i)$. The new batch is accepted to queue j if and only if $x_j + k \leq C_j$. Otherwise the whole batch is rejected.
Similarly, a departure in queue i, $1 \leq i \leq n$ of a batch of size ℓ can only happen if $x_i \geq \ell$.

The arrival and the departure of a batch in a queue are not monotone events. Consider the simple example of a single queue of size C and two states $x = C - 1$ and $x' = C - 2$ respectively. With an event e corresponding to an arrival of a batch of size 2, one gets $\Phi(x, e) = x$ and $\Phi(x', e) = C$, so that $x > x'$ and $\phi(x, e) < \phi(x', e)$, contradicting the monotonicity property.

Once the transition function for the Markov chains describing the behavior of a network with batches is defined, the next question is whether EPSA converges with probability one. The case of a single queue is studied in full details in Appendix. Based on this, the following can be shown for networks.

PROPOSITION 1. *EPSA couples almost surely if and only if batch arrivals or departures of size one happen with positive probability in each queue.*

PROOF. Let p be the probability that batch arrivals or departures of size one happen in each queue. Then, under this type of events only, the network has no batches and is monotone so that envelopes couple. Conversely, if at least one queue Q_i has no arrival and no departure of single customers, then the gap between the two envelopes in Q_i cannot become smaller than the smallest batch size (see Proposition 3 in Appendix). □

If individual customers cannot enter or leave one queue, the regular PSA using function Φ converges almost surely as soon as the batches sizes in each queue have no common divisor and the capacity of the queue is large enough (see Theorem 3 in Appendix). So this is a case where PSA can be applied and EPSA cannot. Section 5 presents a patch for EPSA in this case.

PROPOSITION 2. *The envelopes $\overline{\Phi}(x, y, e)$ and $\underline{\Phi}(x, y, e)$ can be constructed for any couple of states (x, y), $x \geq y$ in time $O(N \log(C))$.*

The proof is given in Appendix B. According to this result, the supremum and the infimum computing cost can be considered negligible (logarithmic in the state space size).

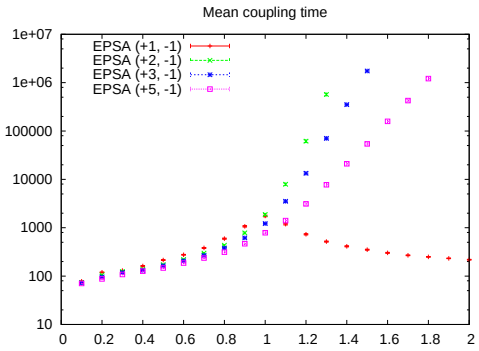

Figure 5: Coupling time in a $(+k, -1)$ queue with batch arrivals of size $k = 1, 2, 3$ and 5 respectively, as a function of the arrival rate λ.

The coupling time is a more difficult point. Here, the coupling time of EPSA can either be comparable or be much larger than the coupling time of PSA, depending on the load of the queues. This is illustrated by the set of experiments displayed in Figure 5. EPSA is used to simulate a single queue with capacity 50 with batch arrivals with rate λ, all with the same size k, and individual departures (denoted a $(+k, -1)$ queue). In that case coupling of EPSA is guaranteed. When the queue is monotone ($k = 1$). the coupling time is always small and is maximum when the arrival rate λ equals the service rate μ. When the queue is not monotone ($k > 1$) the coupling time of EPSA is similar to the monotone case (even marginally smaller) for small values of λ, but grows extremely fast (even on a logarithmic scale) when $k\lambda/\mu$ becomes larger than 1. It is straightforward to explain why this happens. Envelopes can only couple when the

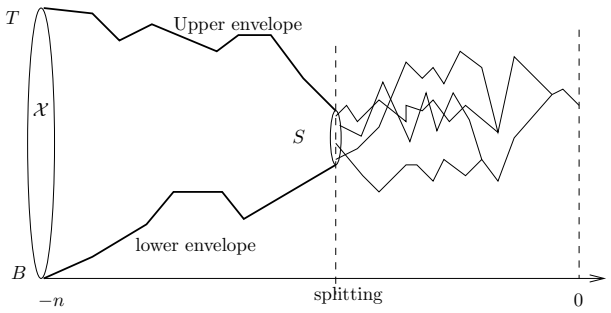

Figure 6: Splitting.

queue becomes empty (see the proof of Proposition 1).When the load grows, this happens with a very small probability. Actually, the classical PSA has similar coupling times, for the same reasons. Again, using EPSA is a clear gain over PSA, the complexity being roughly divided by $|\mathcal{X}|$. Nevertheless, its simulation time may be too large to be used in practice in some cases.

5. BEYOND ENVELOPES

As mentioned before, there are cases where EPSA does not couple or where its coupling time is too large. In that case envelopes can still be useful.

5.1 Splitting and bounds

Even if the two envelopes $\overline{\Phi}, \underline{\Phi}$ cannot couple, they still may get close. Let L be the smallest possible cardinality of the set $\{x \in \mathcal{X} : \underline{\Phi}^n(T, B, e_{0\to n}) \preceq x \preceq \overline{\Phi}^n(T, B, e_{0\to n}\}$.

One way to exploit the envelopes in that case is to continue the simulation once the gap between the envelopes reaches L using a classical PSA, simulating the chain starting with all states between the upper and lower trajectories. This is called splitting in the following.

Algorithm 3: EPSAWS (EPSA With Splitting).

Data: - Φ, $\{e_{-n}\}_{n\in\mathbb{N}}$, Γ
- L size of the set that triggers the splitting
Result: A state $x^* \in \mathcal{X}$ generated according to the stationary distribution of the system
begin
  $n = 1$; $M := T$, $m := B$;
  $S = \{x \in \mathcal{X}, m \le x \le M\}$;
  repeat
    $i := n$;
    repeat
      $(M, m) := \Gamma(M, m, e_{-i})$;
      $i := i - 1$; $S = \{x \in \mathcal{X}, m \le x \le M\}$;
    until $|S| = L$ *or* $i = 0$;
    for $j = i$ **downto** 1 **do** $S := \Phi(S, e_{-i})$;
    $n := 2n$;
  until $|S| = 1$;
  return S;
end

Figure 6 illustrates a successful run of EPSA with splitting (called EPSAWS).

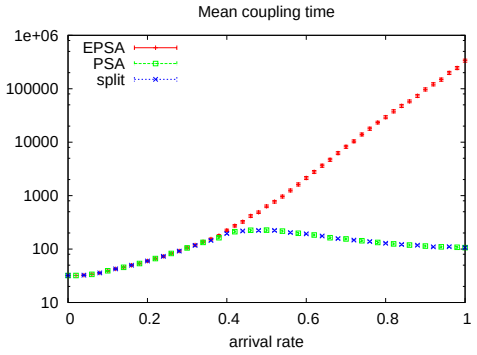

Figure 7: Coupling times for PSA, EPSA and EPSA with splitting for a $(+2, +3, -1)$ queue.

The main use of EPSAWS is when the state space is too big to use PSA directly but envelopes can reduce the size of the space very fast.

Figure 7 reports the coupling times of EPSA, PSA and EPSAWS for a queue with capacity 20 and arrival batches +2 and +3 with probabilities 0.49 and 0.51 and departures of size one (with rate 1). In this queue, Theorem 3 says that all three algorithms converge. In the figure, the coupling time of PSA and EPSAWS are the same while EPSA's coupling time grows very fast when λ/μ is larger than one. In this case EPSAWS has a clear advantage over the other two algorithms.

Another possibility is to continue to compute the envelopes after the spliiting point instead of all trajectories. When time zero is reached, we get M_0 and m_0 that satisfy

$$\mathbb{P}(M_0 \preceq x) \le \Pi(x) \le \mathbb{P}(m_0 \preceq x),$$

where $\Pi(x)$ is the cumulative stationary distribution function (cdf) of the Markov chain: $\Pi(x) = \sum_{y\preceq x} \pi_y$. Note that the gap between M_0 and m_0 may be larger than L. Hopefully, it is still small on most trajectories.

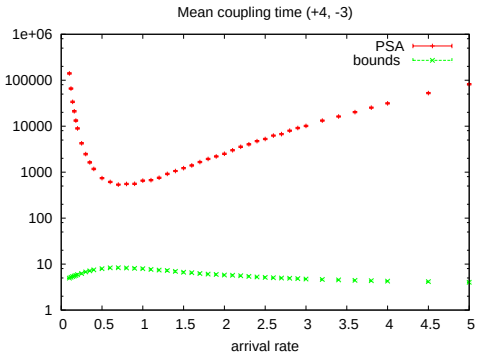

Figure 8: Exact values versus bounds.

Figure 8 reports the coupling times of EPSAWS and the number of steps needed for envelopes to get closer than 4

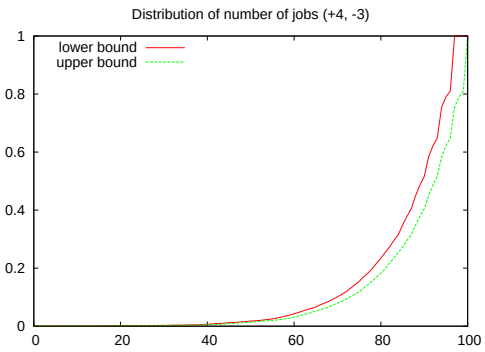

Figure 9: Bounds for the c.d.f. for a $(+4,-3)$ queue.

in a $(+4,-3)$ queue with capacity 15. The theory says that EPSA does not couple here but PSA and EPSAWS will terminate. The Figure shows that the coupling time of EPSAWSis too large to be useful for large queues. On the other hand, the time needed for both envelopes to get close is very small. It allows one to get rather tight upper and lower bounds on the c.d.f. (given for a (+4,-3) queues with capacity 100 and $\lambda/\mu = 1$ in Figure 9).

5.2 Alternative to splitting: change the events

Until now, we supposed that the constructive definition of the Markov chain $X_{n+1} = \Phi(X_n, e)$ is fixed. Taking an alternative description of the chain can reduce the coupling time significantly, or make envelopes converge if this was not the case.

If the Markov chain X_n is irreducible and aperiodic, it is always possible to transform slightly the chain preserving the stationary distribution such that there exists one state 0 for which the transition matrix verifies $P_{00} > 0$. In that case, choose a spanning tree A on the transition graph of the chain rooted in 0 [8]. By defining the order on $\mathcal{X}$ as the distance on that spanning tree A to state 0 and defining an event e which occurs with probability $\min_{(i,j)\in A} P_{ij}$ such that $\Phi(x, e)$ is the next vertex on A after x, and other events in an arbitrary way such that $X_{n+1} = \Phi(X_n, e)$, then EPSA converges with probability one. The reason is that under event e, the chain is monotone and both envelopes converge to 0 in that case.

The main drawback of this approach is that the construction of the spanning tree has a complexity $O(|\mathcal{X}|^2)$ which may be too large in practical cases.

Modifying the event can also be used to improve the coupling time of both EPSA and PSA. We illustrate this on the example of a single queue with batch arrivals and services. In the appendix, we consider the *natural* events to construct the function Φ: *batch arrivals* of size k (event a) and *batch services* of size ℓ (event d). If $\gcd(k, \ell) = 1$ and $k + \ell \leq C + 1$, then Lemma 1 implies that the underlying Markov chain is irreducible and a spanning tree rooted in state 0 can be constructed. Suppose that arrival rate λ and service rate μ verify $\lambda \leq \mu$. Furthermore, without loss of generality, suppose that $\lambda + \mu = 1$. An alternative event description (we denote the corresponding transition function by Ψ) is given by:

Event e_1: For each $x \in \mathcal{X}$, $\Psi(x, e_1) = y$ only if the arc (x, y) is a part of one shortest path to 0. (If there is more than one shortest path from x to 0, then chose one of them.) The rate of event e_1 is set to λ.

Event e_2: For each $x \in \mathcal{X}$, $\Psi(x, e_2) = \Phi(x, d)\delta_{\Psi(x,e_1)=\Phi(x,a)} + \Phi(x, a)\delta_{\Psi(x,e_1)=\Phi(x,s)}$. The rate of event e_2 is λ.

Event e_3: For each $x \in \mathcal{X}$, $\Psi(x, e_3) = \Phi(x, d)$. The rate of event e_3 is $\mu - \lambda$. (If $\lambda = \mu$, there is no event e_3.)

Note that we might have more than one choice for event e_1. However, once we have fixed e_1, then e_2 and e_3 are completely defined. In Table 1 we report simulation times for example with $k = 3$, $l = 2$ and different values of capacity C. We can notice that alternative event description decreases significantly the coupling time. However, for this new event description computing of envelopes becomes more difficult.

C	p=1/4		p=1/3		p=1/2	
	nat.	alt.	nat.	alt.	nat.	alt.
5	150.46	9.54	92.29	13.06	62.28	44.28
10	244.64	16.94	142.51	25.15	187.45	81.56
20	4113.89	31.12	696.30	54.92	1304.00	585.19

Table 1: Coupling times for natural and alternative event description for case: $k = 3$, $l = 2$.

5.3 Computing envelopes

There is a compromise between simulation time and the complexity due to the number of trajectories that need to be considered. As seen in the previous paragraph, an alternative event description can decrease the coupling time of the PSA. However, in this case envelopes become more difficult to compute. One can relax further the notion of envelopes and, instead of taking infimum and supremum states for all the trajectories, one might prefer taking a state that is even smaller (resp. greater) than infimum (resp. supremum) state, but that is much easier to compute, as in the negative customer case.

The question of finding an event representation which is better than the natural event representation, but for which envelopes can be computed efficiently (without generating the state space) remains open for the general case of networks of queues.

6. REFERENCES

[1] A. Bouillard and B. Gaujal. Backward coupling for perfect simulation of free-choice nets. *Journal of Discrete Event Dynamics Systems, theory and applications*, 2006. Special issue of selected papers from the Valuetools conference.

[2] A. Bouillard and B. Gaujal. Backward coupling in petri nets. In *Valuetools*, Pisa, Italy, 2006.

[3] J. Dopper, B. Gaujal, and J.-M. Vincent. Bounds for the coupling time in queueing networks perfect simulation. In *Numerical Solutions for Markov Chain (NSMC06)*, pages 117–136, Charleston, June 2006. Celebration of the 100th anniversary of Markov.

[4] J.-M. Fourneau, I. Y. Kadi, N. Pekergin, J. Vienne, and J.-M. Vincent. Perfect Simulation and Monotone

Stochastic Bounds. In *2nd International Conference Valuetools'07*, Nantes, France, October 2007.

[5] B. Gaujal and F. Perronnin. Coupling from the past in hybrid models for file sharing peer to peer systems. In *Proceedings of the 10th International Conference HSCC'07*, Pisa, Italy, April 2007.

[6] J. G. Propp and D. B. Wilson. Exact sampling with coupled markov chains and applications to statistical mechanics. *Random Structures and Algorithms*, 9(1-2):223–252, 1996.

[7] J.-M. Vincent. Perfect simulation of monotone systems for rare event probability estimation. In *Winter Simulation Conference*, Orlando, Dec. 2005.

[8] J.-M. Vincent and C. Marchand. On the exact simulation of functionals of stationary markov chains. *Linear Algebra and its Applications*, 386:285–310, 2004.

[9] J.-M. Vincent and J. Vienne. Perfect simulation of index based routing queueing networks. *Performance Evaluation Review*, 34(2):24–25, 2006.

[10] J.-M. Vincent and J. Vienne. Psi2 a software tool for the perfect simulation of finite queueing networks. In *QEST*, Edinburgh, Sept. 2007.

APPENDIX

A. A SINGLE QUEUE WITH BATCHES

This appendix is dedicated to the perfect simulation of a single queue with batch arrival and services. This appendix actually contains the main technical difficulties of the paper.

Consider a single queue with finite capacity C, batch arrivals and batch services. Suppose that the arrival batch size is always equal to k and service batch size is equal to ℓ. Function Φ is based on two types of events: arrivals of k customers simultaneously (a_k) and departures of ℓ customers (d_ℓ).

$$\Phi(x,a_k)=\begin{cases} x, & x>C-k \\ x+k, & x\le C-k \end{cases} \quad \Phi(x,d_\ell)=\begin{cases} x, & x<\ell \\ x-\ell, & x\ge \ell. \end{cases}$$

The envelopes are easy to compute (the complexity is the same as for Φ):

$$\overline{\Phi}(M,m,a_k)=\begin{cases} (M+k)\wedge C, & m\le C-k \\ M, & m>C-k \end{cases}$$

$$\underline{\Phi}(M,m,a_k)=\begin{cases} m+k, & M\le C-k \\ (m+k)\wedge((C+k-1)\vee m), & M>C-k \end{cases}$$

$$\underline{\Phi}(M,m,d_\ell)=\begin{cases} (m-\ell)\vee 0, & M\ge \ell \\ m, & M<\ell \end{cases}$$

$$\overline{\Phi}(M,m,d_\ell)=\begin{cases} M-\ell, & m\ge \ell \\ (M-\ell)\vee((\ell-1)\wedge M), & m<\ell. \end{cases}$$

PROPOSITION 3. *If $\min\{k,\ell\}>1$, then the envelopes do not couple (EPSA never converges).*

PROOF. Suppose that envelopes couple in finite time t. Then at time $t-1$ we have $m_{t-1}<M_{t-1}$ and $m_t=M_t$, where $m_t=\underline{\Phi}(M_{t-1},m_{t-1},a_k)$ and $M_t=\overline{\Phi}(M_{t-1},m_{t-1},a_k)$. By symmetry of the problem, without loss of generality we can suppose that the coupling occurs after an arrival. We have three different cases:

- $M_{t-1}+k\le C$. Then $m_t=m_{t-1}+k<M_{t-1}+k=M_t$, which is not possible since we supposed that t is a coupling time.
- $m_{t-1}+k>C$. Then $m_t=m_{t-1}<M_{t-1}=M_t$, which is also not possible.
- $M_{t-1}+k>C$ and $m_{t-1}+k\le C$. Then consider states $x=M_{t-1}$ and $y=x-1\ge m_{t-1}$. Then $\Phi(x,a_k)=x$ and $\Phi(y,a_k)=\begin{cases} x-1, & x>C-k+1 \\ x-1+k, & x\le C-k+1 \end{cases}\neq x$, so we obtain again that $m_t<M_t$.

Therefore, the envelopes do not couple. □

LEMMA 1. *The Markov chain is irreducible if and only if $\gcd(k,\ell)=1$ and $k+\ell\le C+1$.*

PROOF. We show first that both conditions are necessary. If $k+\ell>C+1$, then state $\ell-1$ is absorbing. Indeed, $\Phi(\ell-1,a_k)=\ell-1$ since $\ell-1+k>C$, and $\Phi(\ell-1,d_l)=\ell-1$. If $k+\ell\le C+1$, then either an arrival or a departure modify the state of the system.

Suppose that $\gcd(k,\ell)=d$. Then starting from a state x we can only go to states $y\in[0,C]$ of form $ak+b(-\ell)$. Let $k=sd$ and $\ell=rd$. Then $y=ak+b(-\ell)=(as-br)d$. If $d>1$, then for instance from state 0 we can never reach state 1.

In order to show that these constraints are also sufficient, we will show that if $\gcd(k,\ell)=d$ and $k+\ell\le C+1$, then from each state $0\le x\le C-d$ we can reach state $x+d$ and from each state $d\le x\le C$ we can reach state $x-d$. Then if $d=1$ the chain is clearly irreducible. From Bézout's lemma, there exist $\alpha,\beta\in\mathbb{N}$ such that $\alpha k-\beta\ell=d$. Then state $x+d$ can be reached from state x by a sequence of α arrivals and β services as follows:

While $(\alpha+\beta>0)$ repeat:

- if ($\alpha>0$ and $x+k\le C$) then $\{\alpha\leftarrow\alpha-1,\ x\leftarrow x+k\}$
- else $\{\beta\leftarrow\beta-1,\ x\leftarrow x-\ell\}$

Note that if $\alpha+\beta>0$, then $\alpha=0$ or $x+k>C$ imply $\beta>0$ and $x-\ell\ge 0$. At each iteration $\alpha+\beta$ is thus decreased by 1, so the above algorithm terminates in finite time. State $x-d$ can be reached similarly by a sequence of α' arrivals and β' departures such that $\alpha' k-\beta'\ell=-d$. □

Lemma 1 implies that it is possible to construct a new function Φ such that Algorithm 1 converges. The following result shows that this is also the case with the original function Φ based on arrivals and departures.

LEMMA 2. *PSA converges almost surely if and only if $\gcd(k,\ell)=1$ and $k+\ell\le C+1$.*

PROOF. The proof that the constraints are necessary follows from Lemma 1. We will show that they are also sufficient. Consider two trajectories starting in points at distance $d>0$. It is sufficient to show that we can decrease their distance by 1. Indeed, applying the same reasoning for any two points we can construct a sequence of events for which the trajectories starting at that two points couple, decreasing thus the total number of trajectories by 1. Thus

we can eventually construct a sequence of events for which all trajectories couple. Suppose that $\ell \leq k$. (The other case is symmetrical.) In order to show that we can decrease a distance of two trajectories by 1 we will use the following facts:

1- If $d \geq \ell$, then we can decrease distance by ℓ. We will call this operator A:

$$A(d) = d - \ell,\ d \geq \ell.$$

Suppose that initial positions of two considered trajectories are points x and $x + d$, where $0 \leq x \leq C - d$. Then an event sequence corresponding to operator A is given by $\lceil \frac{x}{\ell} \rceil$ successive departures.

2- If $d < \ell$, then there exists a sequence of events corresponding to operator B:

$$B(d) = \ell - d,\ d < \ell.$$

For an initial configuration $(x, x + d)$ start first by applying $\lfloor \frac{x}{\ell} \rfloor$ successive departures. This gives a new position $(y, y + d)$ such that $0 \leq y < \ell$. If $y + d \geq \ell$, then apply one more departure which modifies distance to $\ell - d$. Thus, an event sequence corresponding to operator b consists of $\lfloor \frac{x}{\ell} \rfloor + 1$ departures. If $y + d < \ell$, then we will show that there is an event sequence that leads to a new configuration $(z, z + d)$ such that one of the following is true: $\big(z < \ell$ and $z + d \geq \ell\big)$ or $\big(z = y+1\big)$. By repeating this a finite number of times we will end up in a configuration $(w, w + d)$ such that $w < \ell$ and $w+d \geq \ell$ from which a departure leads to a configuration with a distance equal to $\ell - d$. We proceed as follows:

- $i = 0$; $y_0 = y$.
- Until z is not found do:
  - $i \leftarrow i + 1$.
  - Use one arrival followed by a finite number of departures in order to reach a new configuration $(y_i, y_i + d)$ such that $y_i < \ell$. Note that the arrival does not modify d since $y_{i-1} + d < \ell$ and $\ell + k \leq C + 1$. Similarly, for a configuration $(u, u + d)$ such that $u \geq \ell$ a departure does not modify d.
  - If $y_i + d \geq l$, then $z = y_i$ and we are in case (a).
  - Else if $y_i = y+1$, then $z = y_i$ and we are in case (b).

We will show that the above algorithm terminates after a finite number of steps. From Bézout's lemma it follows that there exist $\alpha, \beta \geq 0$ such that $\alpha(-\ell) + \beta k = 1$. Consider the configuration $(y_\beta, y_\beta + d)$. We have applied exactly β arrivals and a finite number η of departures. Thus $y_\beta = y + \eta(-\ell) + \beta k$. Now $0 \leq y_\beta < \ell$ and $\alpha(-\ell) + \beta k = 1$ imply $\eta = \alpha$.

3- If $d < \ell$, then similarly to the above case, there exists a sequence of events corresponding to operator D:

$$D(d) = k - d,\ d < \ell.$$

(We only need $d < k$ to get the existence of a corresponding sequence, but we will always apply operator a when $d \geq \ell$.)

From Bézout's lemma it follows that there exist $\gamma, \delta \geq 0$ such that $\gamma(-\ell) + \delta k = -1$. A sequence of events allows to decrease the distance by 1 can be then obtained as follows. We suppose that initially $d < \ell$ (otherwise we can apply operator A):
1- Apply δ times operator $B \circ D$. After each step make sure that we still have $d < \ell$. If not, apply operator A as many times as needed to have $d < \ell$. This gives $d' = \delta(k - \ell) - \zeta\ell + d$.
2- Remark that $\zeta = \gamma - \delta$. Thus $d' = d - 1$. □

Consider now a general case with arrival batch sizes that belong to a set $\mathcal{K} = \{k_1, k_2, \ldots, k_K\}$ and service batch sizes that belong to a set $\mathcal{L} = \{\ell_1, \ell_2, \ldots, \ell_L\}$. Without loss of generality we can assume that $k_1 < \ldots < k_K$ and $\ell_1 < \ldots < \ell_L$. Denote by $\mathcal{M} = (\mathcal{L} \cup \mathcal{K}) \backslash \{\ell_1, k_1\} = \{m_1, m_2, \ldots, m_M\}$, with $m_1 < \ldots < m_M$. Note that $M \leq K + L - 2$. Let $g_0 = \gcd(k_1, \ell_1)$ and

$$g_i = \gcd(g_{i-1}, m_i),\ 1 \leq i \leq M.$$

Furthermore, we suppose that $g_i > 1$, for all $i < M$. If for $i < M$ we have $g_i = 1$, then we can consider $\mathcal{M}' = \{m_1, \ldots, m_i\}$ instead of $\mathcal{M}$.

Proposition 4. *The Markov chain is irreducible if* $\gcd(\mathcal{L} \cup \mathcal{K}) = 1$ *and*

$$\max\Big\{k_1 + \ell_1,\ \max_{i=1}^{M}(g_{i-1} + m_i)\Big\} \leq C + 1. \qquad (1)$$

Proof. The proof that the above conditions are sufficient is similar to the proof of Lemma 1. Indeed, only by arrivals of size k_1 and services of size ℓ_1 we can reach all the states of the same equivalence class modulo g_0. Suppose now that m_i corresponds to an arrival (the other case is symmetrical). Then condition $g_{i-1} + m_i \leq C + 1$ implies that from each state x we can go either to a state $x - g_{i-1} \geq 0$ or to a state $x + m_i \leq C$. Then similarly to the proof of Lemma 1, it can be shown that from a state x we can reach all the states of the same equivalence class modulo $g_i = gcd\{g_{i-1}, m_i\}$. Since $g_M = 1$, the chain is irreducible. □

Similarly to the proof of Lemma 1, the constraint $\gcd(\mathcal{L} \cup \mathcal{K}) = 1$ is also necessary, since otherwise starting from state x we can only visit states of type $y = x+\lambda d$, where $\lambda \in \mathbb{Z}$ and $d = \gcd(\mathcal{L} \cup \mathcal{K})$. Condition $k_1 + \ell_1 \leq C + 1$ is also necessary (otherwise state $\ell_1 - 1$ is absorbing). The other conditions are not always necessary. Take for instance $\mathcal{K} = \{4, 11\}$, $\mathcal{L} = \{8, 10\}$ and $C = 12$. Then it can be easily verified that the chain is irreducible. On the other hand, $g_0 = 4$, $m_1 = 10$, but $g_0 + m_1 = 14 > 13 = C + 1$.

Theorem 3. *PSA converges almost surely if* $\gcd(\mathcal{L} \cup \mathcal{K}) = 1$ *and* $C \geq 3\max(\mathcal{L} \cup \mathcal{K})$.

Proof. Suppose that $\ell_1 \leq k_1$ (the other case is symmetrical). Then we can find a sequence of events corresponding to operators A and $B_m,\ m \in \{\ell_1, k_1\} \cup \mathcal{M}$:

$$\begin{aligned} A(d) &= d - \ell_1,\ d \geq \ell_1, \\ B_m(d) &= m - d,\ d < \ell_1,\ m \in \{\ell_1, k_1\} \cup \mathcal{M}. \end{aligned}$$

Indeed, for two trajectories starting in points x and $x + d$, an event sequence corresponding to operator a is given by $\lceil \frac{x}{\ell_1} \rceil$ events d_{ℓ_1}. For operators B_m the event sequence is slightly more complex, but follows similar steps as in the proof of Lemma 2. Under condition $C \geq 3\max_{m \in \mathcal{L} \cup \mathcal{K}} m$,

it is always possible to translate the two points by 1 to the right (resp. left) using Bézout's lemma, without hitting the boundaries (i.e. without modifying distance d). Thus for any $\ell \in \mathcal{L}$ (resp. $k \in \mathcal{K}$) and any two initial states x and $x+d$ we can always reach a state $(y, y+d)$ such that $y < \ell$ and $y+d \geq \ell$ (resp. a state $(z, z+d)$ such that $z \leq C-k$ and $z+d > C-k$) with a finite sequence of events. Then applying an additional event d_ℓ (resp. a_k) yields a sequence of events corresponding to operator b_ℓ (resp. b_k).

Bézout's lemma implies that there are $\alpha \geq 0$ and $\beta_x \in \mathbb{Z}$, $x \in \mathcal{M} \cup \{k_1\}$ such that:

$$\alpha(-\ell_1) + \sum_{m \in \mathcal{M} \cup \{k_1\}} \beta_m m = -1.$$

Suppose that we have two trajectories at distance $d < \ell_1$ (otherwise use operator a). Then a sequence of events that decreases distance by 1 can be obtained as follows:

For each $m \in \mathcal{M} \cup \{k_1\}$ apply $|\beta_m|$ times:

- a sequence of events corresponding to operator $B_{\ell_1} \circ B_m$ if $\beta_m > 0$, and $B_m \circ B_{\ell_1}$ if $\beta_m < 0$,
- a sequence corresponding to a finite number of successive applications of operator A, such that after each iteration we have a new distance $d < \ell_1$.

This gives a new distance

$$d' = d + \sum_m \beta_m (m - \ell_1) + \gamma(-\ell_1) < \ell_1.$$

Finally, remark that $\alpha = \sum_m \beta_m + \gamma$. Thus, $d' = d - 1$. □

B. PROOF OF PROPOSITION 2

Here is the construction of $M' = \overline{\Phi}(M, m, a_k)$ and $m' = \underline{\Phi}(M, m, a_k)$ when a batch of k customers are routed into the system according to the index functions I. It should be clear looking at the construction of M' and m' below that it can be done in $O(N \log C)$, where C is the maximal capacity of all queues by using a dichotomic search.

For each queue i we first introduce for notational convenience, two functions $h^{(i)}$ and $H^{(i)}$, describing the infimum and supremum values for the number of customers in queue i after a batch arrival of size k in queue i, given that the number of customers in that queue just before the new arrival is between u and v:

$$H^{(i)}(v,u) = \begin{cases} (v+k) \wedge C, & u \leq C-k \\ v, & u > C-k \end{cases}$$

$$h^{(i)}(v,u) = \begin{cases} u+k, & v \leq C-k \\ (u+k) \wedge ((C+k-1) \vee u), & v > C-k. \end{cases}$$

Let d_m and d_M be defined as follows:

$$d_m = \arg\min_i I_i(m), \quad d_M = \arg\min_i I_i(M).$$

We consider all the queues $\ell = 1..N$. For each of them, two different cases have to be considered.

Case 1: $I_\ell(m) > I_{d_M}(M)$. Then obviously $\ell \neq d_M$, since $m \leq M$ and I is increasing. For all z, $m \leq z \leq M$, $I_\ell(z) \geq I_\ell(m) > I_{d_M}(M)$ which implies $\Phi_\ell(z, a_k) = z_\ell$, $m \leq z \leq M$, where $\Phi = (\Phi_1, \ldots, \Phi_N)$. Thus,

$$m'_\ell = m_\ell, \quad M'_\ell = M_\ell.$$

Case 2: $I_\ell(m) \leq I_{d_M}(M)$. We consider first M'_ℓ. We have two cases:

- If $\ell = d_M$, then
$$M'_{d_M} = H^{(d_M)}(M_{d_M}, m_{d_M}).$$
Indeed, for state $z = (M_1 \ldots M_{d_M - 1}, a, M_{d_M+1} \ldots M_n)$ we clearly have $\arg\min_i I_i(z) = d_M$.
- If $\ell \neq d_M$, then we consider state:
$$z = (M_1, \ldots, M_{\ell-1}, m_\ell + b, M_{\ell+1}, \ldots, M_N),$$
where $b \geq 0$ is defined as follows:
$$\begin{aligned} I_\ell(m + 1_\ell) &\leq I_{d_M}(M), \\ &\vdots \\ I_\ell(m + b1_\ell) &\leq I_{d_M}(M), \\ I_\ell(m + (b+1)1_\ell) &> I_{d_M}(M). \end{aligned}$$
Then $\forall \alpha \neq \ell$,
$$I_\ell(z) = I_\ell(m + b1_\ell) \leq I_{d_M}(M) \leq I_\alpha(M) = I_\alpha(z),$$
and $\arg\min_i I_i(z) = \ell$. Furthermore, state z is a state with the highest value of the component ℓ among all the states in which the new batch is sent to queue ℓ. Thus for the upper envelope, we have:
$$M'_\ell = \max\{M_l, H^{(\ell)}(m_\ell + b, m_\ell)\}.$$

As for the lower envelope, we have the following cases:

- If $\ell \neq d_m$, then $m'_\ell = m_\ell$.
- If $\ell = d_m$ and $I_{d_m}(M) \leq \min_{i \neq d_m} I_i(m)$ (note that this implies $d_m = d_M$), then:
$$m'_{d_m} = h^{(d_m)}(M_{d_m}, m_{d_m}).$$
- If $\ell = d_m$ and $I_{d_m}(M) > \min_{i \neq d_m} I_i(m)$ then we consider state
$$z = (m_1, \ldots, m_{d_m - 1}, m_{d_m} + c, m_{d_m+1}, \ldots, m_N),$$
where $c \geq 0$ is defined as follows:
$$\begin{aligned} I_{d_m}(m + 1_{d_m}) &\leq \min_{i \neq d_m} I_i(m), \\ &\vdots \\ I_{d_m}(m + c1_{d_m}) &\leq \min_{i \neq d_m} I_i(m), \\ I_{d_m}(m + (c+1)1_{d_m}) &> \min_{i \neq d_m} I_i(m). \end{aligned}$$
Note that $m_{d_m} + c$ is the maximal value of component d_m for which we still always send the new batch to queue d_m. We have:
$$m'_{d_m} = \min\{m_{d_m} + c + 1, h^{(d_m)}(m_{d_m} + c, m_{d_m})\}.$$